Software Development Principles

Additional Notes: Debugging

Lecturer:

Karen Nolan

karen.nolan@it-tallaght.ie

Python Debugging

- As algorithm complexity increases, proportionally finding bugs in algorithms with mathematical concepts and / or processing can be equally complex.
- Sometimes adding simple print statements can help, this allows you to track variable states.
- This adds a time issue and can even lead to its own complexity issues.
- A solution is break points and debugging.
- PyCharm has its own method but very similar to other IDE's such as Visual Studio.

Python Debugging

- Lets use this example of Python operators.

- Tracing the variable states could be complex (even in this basic example)

```
x = 8
y = 9

x = y + 2
y = y % 2

print(x)
print(y)
```

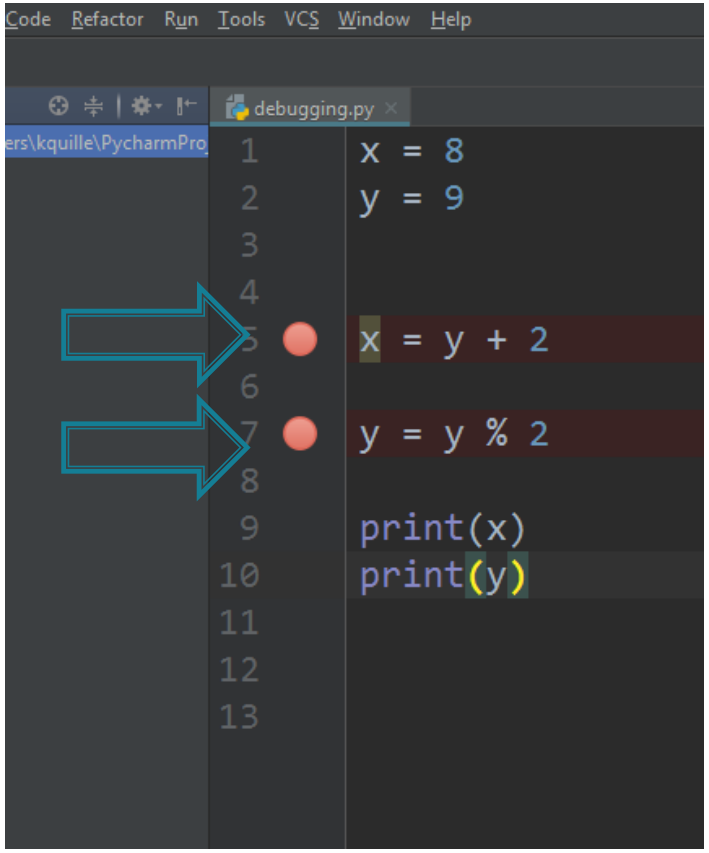
Python Debugging

- A solution perhaps?
- What about scaling this solution?

```
x = 8
y = 9
print(x)
print(y)

x = y + 2
print(x)
print(y)

y = y % 2

print(x)
print(y)
```

Python Debugging

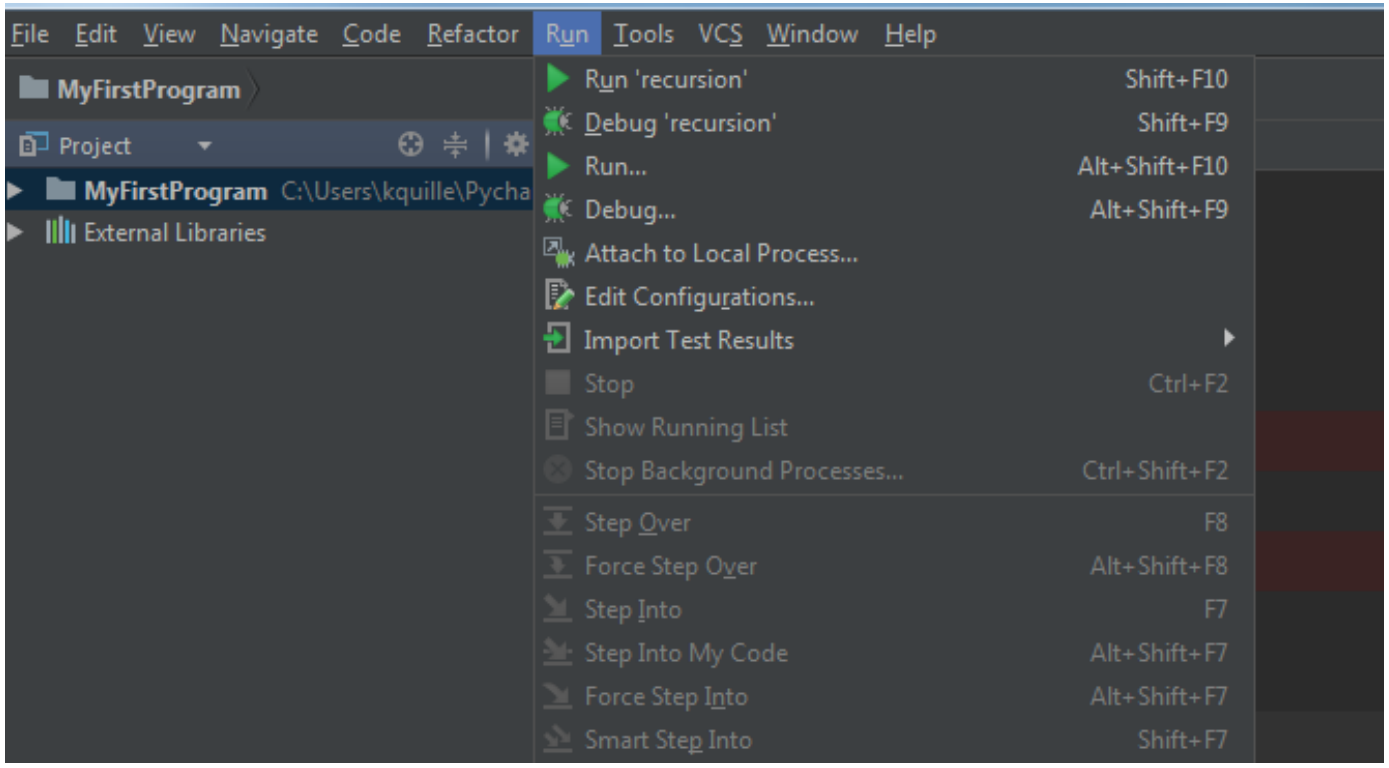
- Create Breakpoints:
- Simply left click in the space between line number and code.
- This tells the debugger where to break.
- A break stops the script and allows you to examine variable states.



(To remove break points, left click again)

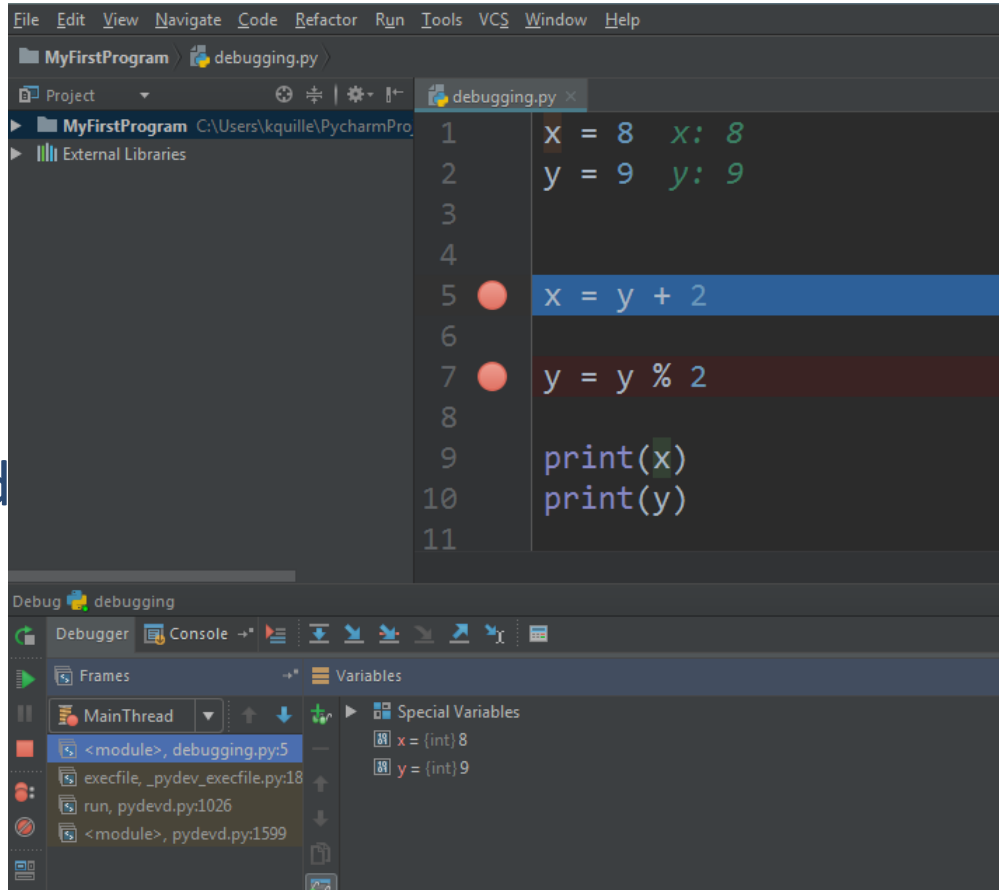
Python Debugging

- Debug:
  - The break points are NOT active during the run command Shift+F10.
  - To debug you must select debug which is Shift+F9



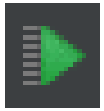
Python Debugging

- Debug:
- You now get a debugger window at the bottom of the IDE.
- Also the debug has stopped at line five (it does not include this line)
- And the variable contents are displayed.

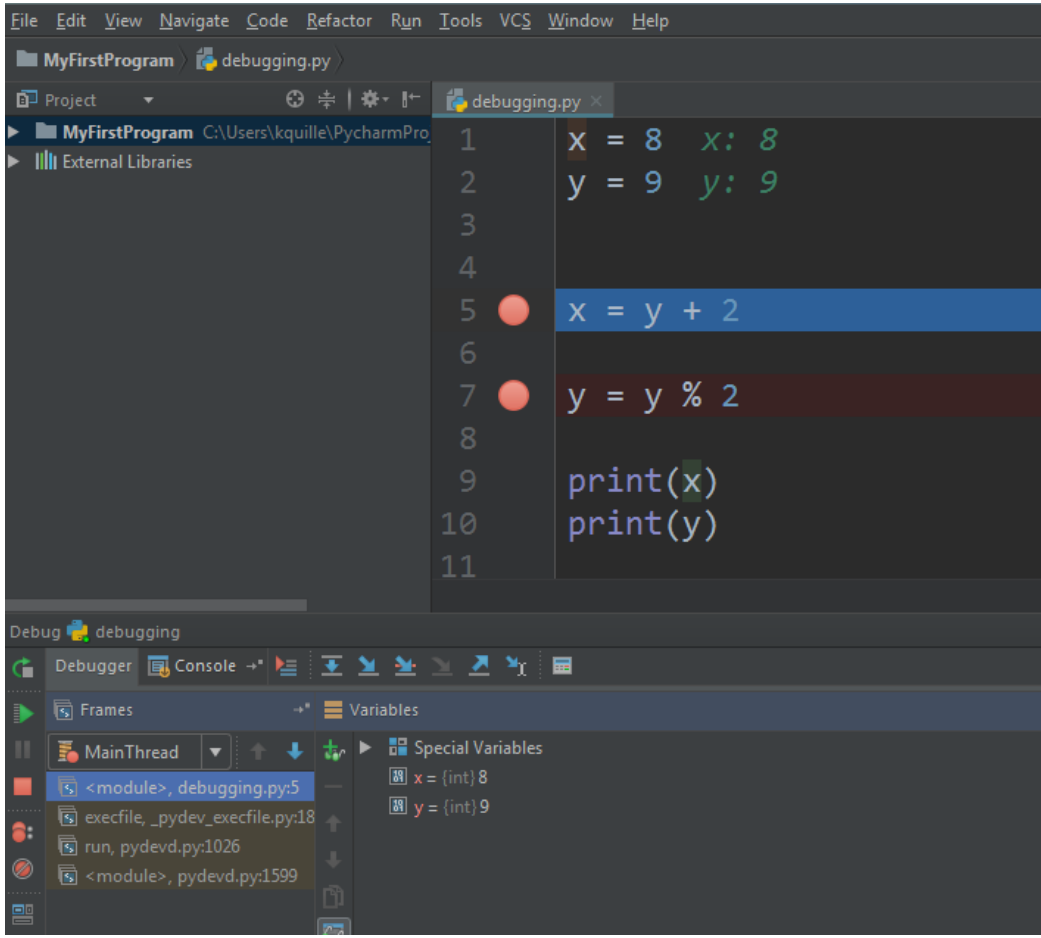


Python Debugging

- Debug:



- This is the continue button. This will move to the next break point or to the end of the script, whichever is first.

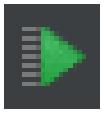


- Notice:
- X= 8
- Y = 9

(it even shows the types)

Python Debugging

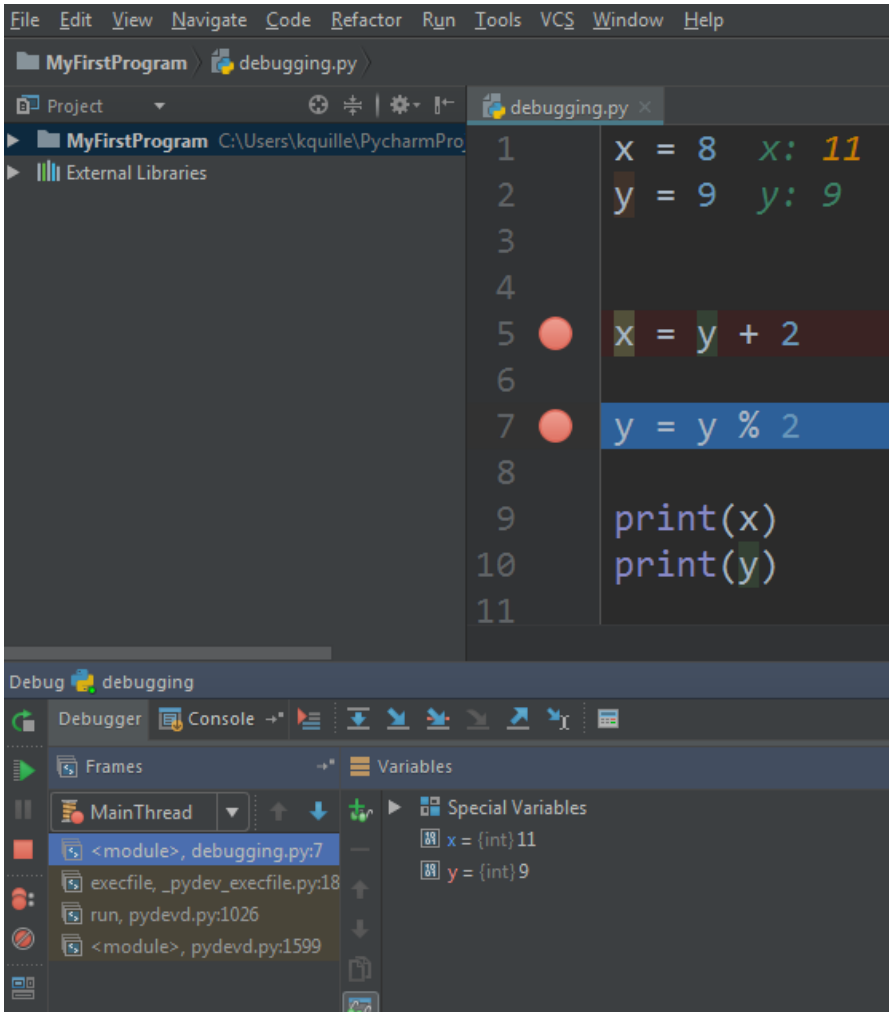
- Debug:



- This is the continue button. This will move to the next break point or to the end of the script, whichever is first.

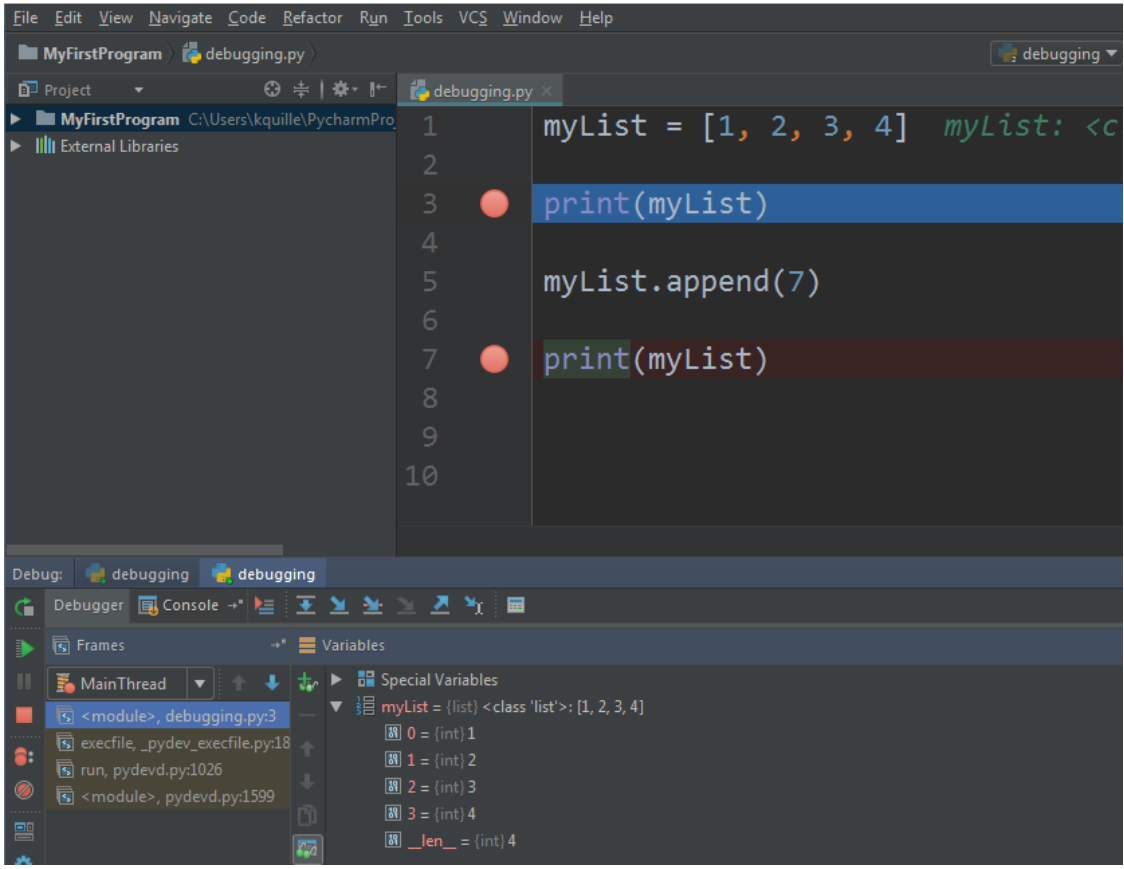
- Notice:
- X= is now 11 (also highlighted on line 1
- Y = 9

(it even shows the types)



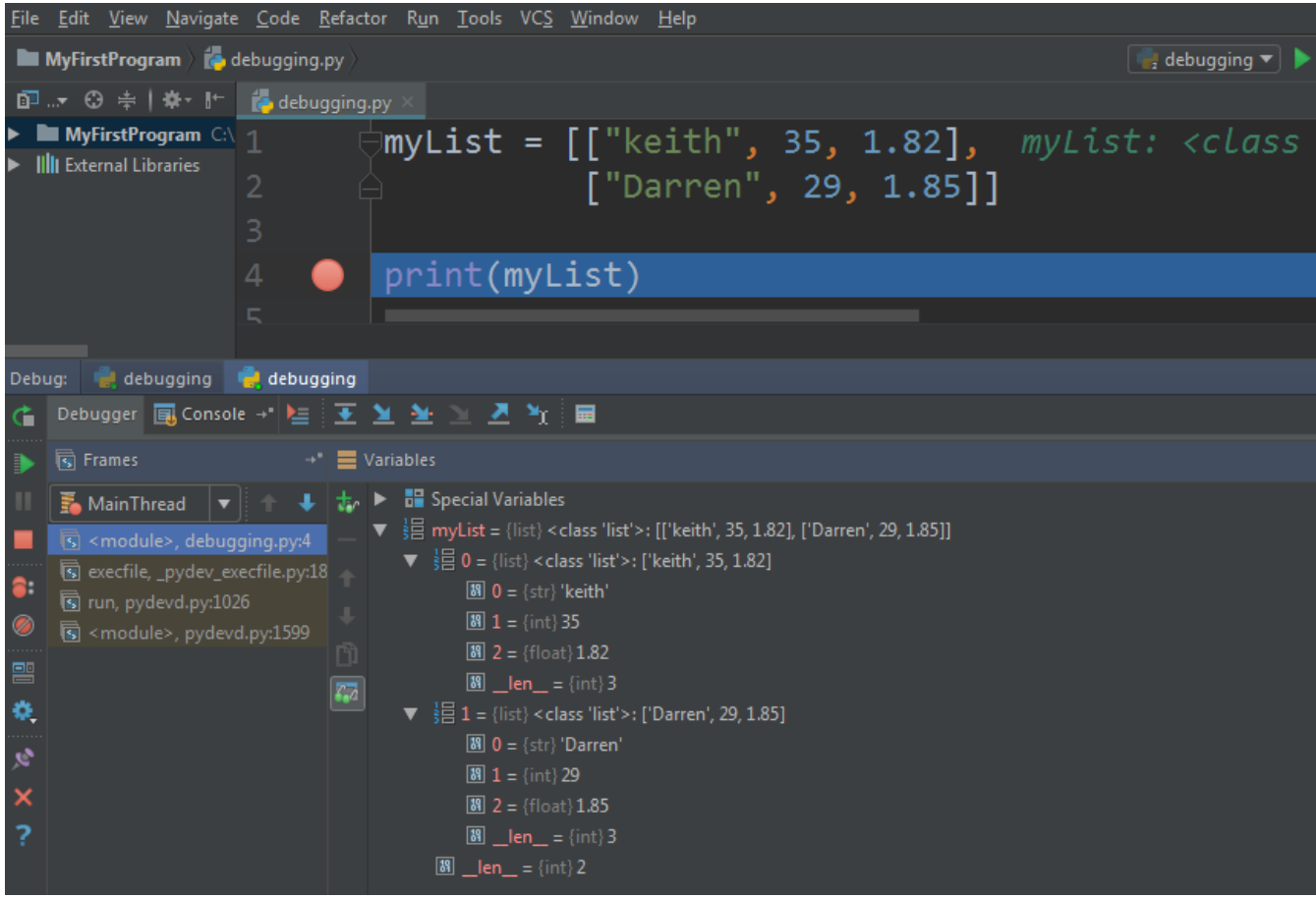
Python Debugging > Lists

- Debug: Lists
- The list can be expanded.
- This allows detailed inspection of the lists contents.
- Very useful tool for 2D lists.



Python Debugging > 2D Lists

- Debug:
- Example of data structure exploration using debug mode in PyCharm



Python Debugging > 2D Lists

- In Class:

- Try examine contents of 2x 1D lists using break points with the following examples:
  - Shallow Copy
  - Deep Copy